Photon Streaming for Interactive Global Illumination in Dynamic Scenes

Daniel Meneveaux, Flavien Bridault

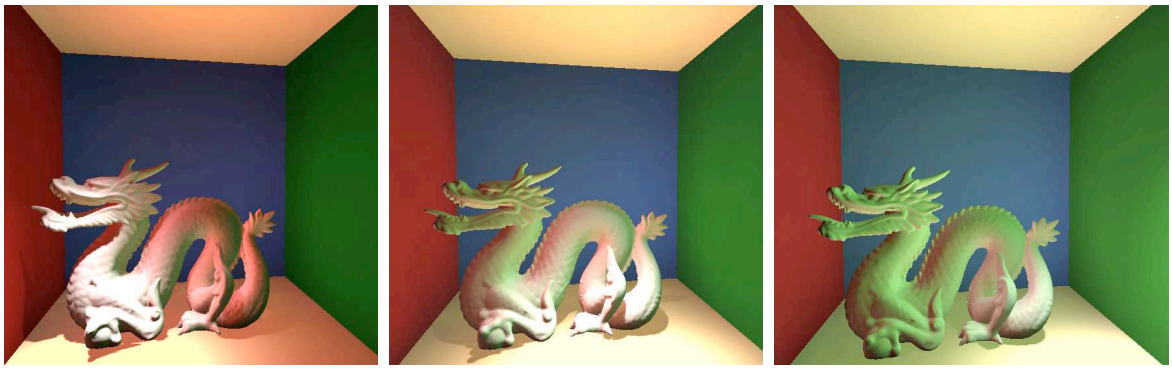
Photon Streaming for Interactive Global Illumination in Dynamic Scenes

Daniel Meneveaux, Flavien Bridault
Laboratoire XLIM, département SIC, Université de Poitiers

October 15, 2008

Abstract

While many methods exist for physically computing diffuse light inter-reflexions, relatively few of them are adapted to dynamic scenes. Despite many approximations made on the formal rendering equation, managing dynamic environments at interactive or real-time frame rates still remains one of the most challenging problem of these years. This paper presents a lighting simulation system based on photon streaming, performed continuously on the CPU. The energy corresponding to each photon impact is associated to regions of space, defined by points and used during the rendering phase as virtual surface element light sources (VSLs). The rendering process can be performed interactively on the GPU, using several approximations. As shown in the results, our method provides high framerates for dynamic scenes, with moving objects or light sources.



(a) (b) (c)

3 images from our photon streaming rendering system: (a) the light source is close to the left red wall; (b) the light source is centered in the scene; (c) the light source is close to the right green wall.

0.1 Introduction

Lighting simulation systems aim at estimating multiple reflections of light in a 3D environment. Even though many authors propose heuristics for reducing computing time, managing moving lights and objects requires the computation of a global illumination solution for each frame.

Our goal is to provide a system that handles dynamic scenes while updating global illumination on the fly. Therefore, our method is based on a continuous stream of photons emitted inside the environment, that can be considered as the light flux emitted by a light source in the real world. For softening flickering during the rendering phase, photon energy is cumulated onto virtual *surface element light sources* (VSLs), placed a priori onto the scene geometry.

In this paper, our contributions include :

- continuous photon shooting in the environment, corresponding to the real world lighting process;
- the accumulation of photons energy on pre-distributed points (placed independently from lighting considerations) in the scene for representing indirect lighting;
- the use of these points as virtual surface element light sources (VSLs from now on) during rendering, for reducing spot artifacts and flickering, instead of virtual *point* light sources (VPLs);
- a method for rendering realistic images at interactive frame rates with dynamic environments, though without taking VSLs shadows into account.

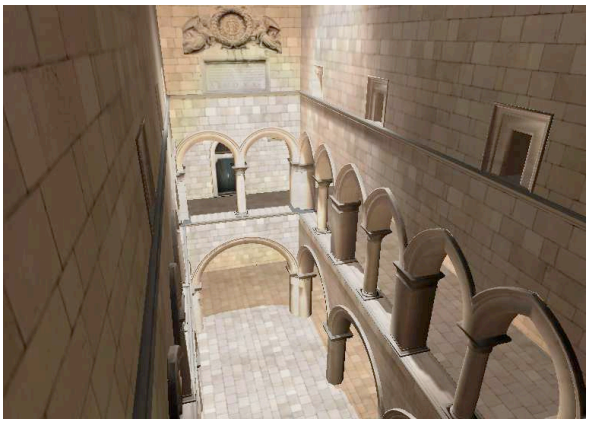


Figure 1: Results of images produced at about 20 frames per second with Sponza Atrium, containing more than 66K triangles and 60K vertices.

This paper is organized as follows. Section 0.2 presents previous work related to (interactive) global illumination in dynamic scenes. Section 0.3 provides an overview of our lighting simulation and rendering system. Section 0.4 details the use of virtual surface light sources. Section 0.5 shows how photon streaming can be used for rapidly updating global illumination in dynamic scenes. Section 0.6 explains the rendering phase and the assumptions we make for reducing computing time. Section 0.7 provides technical aspects about our data structure. Section 0.8 shows the results obtained. Section 0.9 concludes and presents future work.

0.2 Related Work

Though radiosity have been largely employed for computing global illumination, interactively updating dynamic scenes requires (re)meshing for each frame when objects

move. Wavelet radiosity can be effectively used for interactive rendering in static environments with moving light sources and glossy inter-reflexions [KTHS06], but the problem remains difficult for moving objects.

Precomputed Radiance Transfer is often used for estimating reflected light around an object [pSKS02, KAMJ05]. The drawback of this family of methods lies in the pre-computing time and the memory required estimating incoming radiance, at many points around each object, or in the scene, according to many incident light directions.

Radiance or irradiance caching has also successfully been used for global illumination, using temporal gradients [GKBP05], with low-frequency BRDFs when the animation is known a priori. However, as shown in [KG05], interpolating values between samples provide artifacts and should be carefully managed. In addition, computing new caches using Monte-Carlo methods remains time consuming when the viewpoint moves.

Photon tracing approaches [Jen01] are also used as a basis for computing light inter-reflexions. For instance some authors [DBMS02] propose to selectively trace photons into regions that require update, using priority criteria for tracing appropriate photons. Perception is also used for guiding computations. This method implies constructing regions with hierarchical photon shooting, using selective and perception criteria for reducing artifacts. In [LC04], several photon-maps are used for reducing visual artifacts and selecting photons during lighting simulation. This approach also proposes a hierarchical photon tracing system. However, constructing the photon-map remains time-consuming due to balancing and should be reconstructed for each frame when objects move.

Instant radiosity is often used as a basis for providing real-time rendering of complex scenes [WBS03, GWS05]. However, these types of methods need attention because of flickering due to inhomogeneous redistribution of VPLs at each frame. VPLs distribution can also be improved as shown in [LSK*07], and the viewpoint can also be taken into account [SIP07, SIP06]. Using resampling strategies can reduce processing time and/or flickering, but still requires additional processing.

In [NPG03], interactive global illumination is performed using environment maps distributed according to a regular grid. Several bounces of light inter-reflections can be taken into account associating hemispherical harmonics to each location. However, the number of environment maps should be increased according to the scene complexity, also increasing sensibly the rendering time.

In this paper, we propose a lighting simulation system dedicated to diffuse inter-reflexions in dynamic environments, without a priori knowledge. It can be seen as a hybrid approach between photon-tracing and instant radiosity, that naturally solves several drawbacks such as flickering and inter-reflection light spots. No photon map balancing is required due to the use of virtual surface element light sources, that are updated on the fly in $\mathcal{O}(1)$ complexity.

0.3 System Overview

Our lighting simulation system operates as follows:

- points defining virtual surface element light sources (VSLs) are distributed onto the environment surfaces;
- photons are shot continuously and stored in a photon pool of fixed size;
- every time a new photon hits an object, it replaces the oldest photon in the pool (queue)
- the new photon energy is associated with the closest VSL, and the oldest photon energy is subtracted from its associated VSL.

The rendering phase relies on geometry and VSLs. It can be performed either using ray tracing or GPU, but this paper rather focuses on GPU interactive rendering. For each frame, direct shadows are estimated using shadow maps, direct lighting is performed using fragment shaders, and indirect reflections are taken into account using VSLs power (also using fragment shaders). Both direct and indirect illumination can be estimated using deferred shading.

Figure 2 illustrates our main system. VSLs are placed in a few seconds when loading the scene geometry (distribution criteria are discussed in section 0.4). Our programs allows rendering immediately. Photon shooting can be done one time or continuously (photon streaming) so as to update VSLs energy.

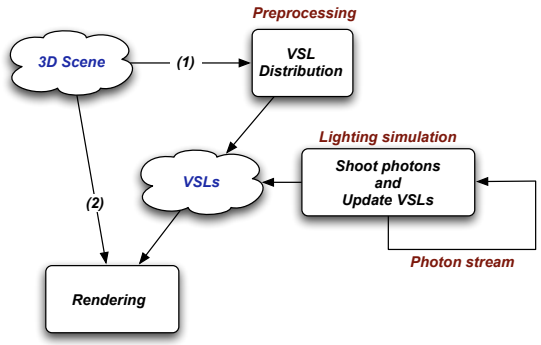


Figure 2: System overview: VSLs are placed in the environment within a few seconds, according to geometry; photon streaming can operate continuously for dynamic global illumination while objects move.

0.4 Virtual Surface Light Sources

Instant radiosity have been used by several authors for describing light interreflexions in diffuse environments [Kel97, WBS03, SIP06, SIP07]. This process produces virtual point light sources (VPLs), which position remains difficult to control due to Monte-Carlo distribution. Light spots may alter the images realism, for instance when a VPL is located in a room corner or close to objects. In addition, for dynamic scenes, lighting simulation implies tracing new samples, producing undesirable flickering artifacts.

0.4.1 VSLs Definition

In this paper, we propose several enhancements that highly reduce these artifacts. First, we do not use VPLs as photons impacts because of flickering. We instead introduce a precomputed repartition of points in the scene that define regions (and also VSLs centers) used for collecting inter-reflections energy. The repartition heuristics are such that these points are located far from polygons edges so as to avoid light spots (for instance in room corners). During rendering, they are not considered as virtual *point* light sources, but as virtual *surface element* light sources, highly reducing light spots.

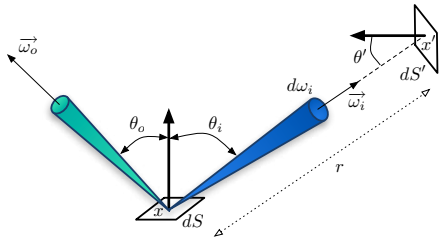


Figure 3: Incident and outgoing light geometry for a given surface element dx.

Let us consider the radiance equation (figure 3):

$$L_o(x,\overrightarrow{\omega_o}) = L_o^e(x,\overrightarrow{\omega_o}) + \int_\Omega L_i(x,\overrightarrow{\omega_i}).R(x,\overrightarrow{\omega_i},\overrightarrow{\omega_o}).cos\theta_i.d\omega_i$$

with

$$d\omega = \frac{dS'.cos\theta'}{r^2}$$

when considering surface elements. Consequently, the term

$$\int_\Omega L_i(x,\overrightarrow{\omega_i}).R(x,\overrightarrow{\omega_i},\overrightarrow{\omega_o}).cos\theta_i.d\omega_i$$

can be rewritten as:

$$\int_{x'\in scene} L'_o(x',\overrightarrow{\omega_i}).R(x,\overrightarrow{\omega_i},\overrightarrow{\omega_o}).V(x,x').\frac{cos\theta_i.cos\theta'}{r^2}dS'$$

In addition, the radiosity exiting from a surface element dS' is: $B' = d\Phi'/dS'$, where $d\Phi'$ is given by the light energy associated with each VSL (Φ_s). Thus, for diffuse surfaces,

$$L'_o(x,\overrightarrow{\omega}) = \frac{B'}{\pi} = \frac{\Phi_s}{dS'.\pi}.$$

In addition, $R(x,\overrightarrow{\omega_i},\overrightarrow{\omega_o})$ is constant and renamed $R(x)$=$\rho(x)/\pi$. $\rho(x)$ being the surface albedo at the given surface element around x and Φ_s being the surface light source energy. Finally, considering that light inter-reflections can be concentrated at VSLs, the integral becomes the following sum:

$$L_o(x,\overrightarrow{\omega_o}) = L_o^e(x,\overrightarrow{\omega_o}) + \sum_{\Phi_s\in VSLs} \Phi_s.R(x).\frac{cos\theta_i.cos\theta'}{\pi.r^2}$$

Note that compared to instant radiosity, this formulation introduces the cosine term $cos\theta'$, depending on the *surface* light source orientation. This orientation can be taken into account using the surface normal corresponding to each VSL. In practice, the normal is stored in the VSL data structure.

0.4.2 VSLs Distribution

VSLs are placed randomly onto the scene surfaces as uniformly as possible, so as to represent the overall global light inter-reflections in the scene. In addition, we wish the pre-processing step be as fast as possible.

The number of VSLs N_v is a user-given parameter of our lighting simulation program. Using the scene total surface area, a VSL density is estimated, providing for each object the number of associated VSLs. Placing VSLs consists in randomly choosing a triangle on the object and (also randomly) placing a point on its surface.

In order to move VSLs away from edges, a small scale is virtually applied to triangles before distribution (see Figure 4.a).

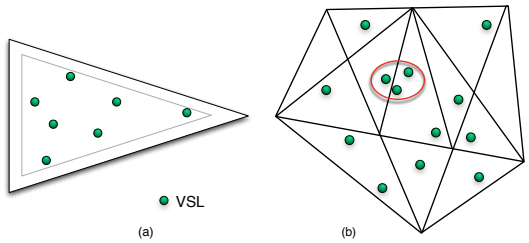


Figure 4: Distribution of VSL points onto scene surfaces: (a) for avoiding accumulation of VSLs on triangle edges, a scale is applied to triangles; (b) the resulting distribution may still contain several VSLs close to one another.

Unfortunately, distributing VSLs on surfaces independently using such a random process may result in placing several VSLs very close to one another (Figure 4.b). This is why we proceed in two steps. The first step consists in increasing the VSLs density (computed from the number given by the user). Experimentally, an increase of 10% provides interesting results. This process generates too many VSLs. The second step consists in iteratively removing close VSLs, until N_v is reached (Figure 5). The naïve process requires to estimate the distance between all the pairs of VSLs points, corresponding to a complexity in $\mathcal{O}(n^2)$ every time a VSL has to be removed.

Instead of computing all the distances at every step, we maintain for each VSL V two distances d_{min1} and d_{min2} corresponding respectively to the two closest VSLs V_1 and V_2 (Figure 5), as detailed in Algorithm 1. d_{min1} defines which is the closest distance between two VSLs while d_{min2} is used to determine which one should be actually removed. This process only requires $\mathcal{O}(n)$ comparisons instead of $\mathcal{O}(n^2)$ for each removed VSL.

Note that: (i) VSLs are attached to the objects, so that when the user moves an object, he also moves the corresponding VSLs; (ii) at least one VSL is attached to

objects; (iii) this process does not actually produce a uniform distribution of VSLs.

Algorithm 1 Additional VSLs removal

for all VSLs V_i **do**
  compute $V_i.d_{min1}$ and $V_i.d_{min2}$ distances;
end for
while $N_{vsls} > N_v$ **do**
  let V_k be the VSL having the smallest d_{min1} and d_{min2} values
  for all VSLs V_i **do**
    if $V_i.V_1 = V_k$ or $V_i.V_2 = V_k$ **then**
      update $V_i.d_{min1}$ and/or $V_i.d_{min2}$
    end if
  end for
end while

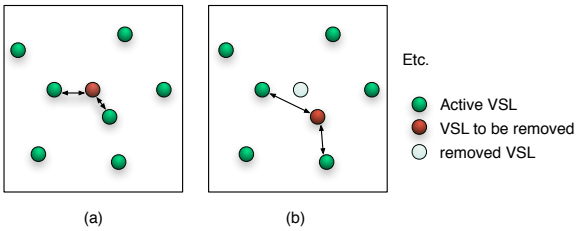


Figure 5: Removing additional VSLs is performed choosing the VSL having the closest distance to its 2 closest neighbors: (a) first VSL removal; (b) removing a second VSL requires updating all the distances d_{min1} and d_{min2}.

0.5 Photon Streaming

Our photon streaming approach simulates light flux continuously emitted in the environment. In practice, instead of re-computing the entire lighting simulation for every frame, we use the VSLs energy corresponding to the photons stored in the pool. The data structure associated with each photon contains a reference to the corresponding closest VSL (figure 6).

Every time a new photon P_n impact is computed, the following operations are performed:

Algorithm 2 New photon P_n impact

- Let V_o be the VSL corresponding to the oldest photon P_o
- $V_o.\Phi$ -= $P_o.\Phi$
- remove P_o from the pool (queue)
- find the VSL V_n closest to P_n
- $V_n.\Phi$ += $P_n.\Phi$

Let us consider a light source emitting a given light power (in Watts), corresponding

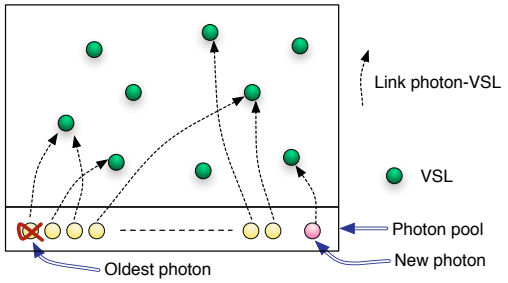


Figure 6: Links between photons and VSLs. When a new photon impact is computed, the photon pool is updated as well as the energy of the VSLs. Each photon has a link to its associated VSL allowing $\mathcal{O}(1)$ complexity update. Note that a VSL can receive light flux from 0 or several photons.

to the energy emitted per time unit. The overall lighting in the scene at a given frame should be represented by the overall energy distributed in the scene.

The photon pool corresponds to an approximation of this energy. Physically, this value should be entirely re-computed for every frame (which is the case for many existing methods). However, the entire photon pool is rapidly updated, producing a slight latency when objects move. In addition, when the viewpoint is changed while the scene remains static, photon streaming can be interrupted.

0.6 Rendering Process

The photon pool is not directly used for rendering so as to avoid splatting noise or long final gather. We instead use VSLs irradiance. We have implemented a GPU rendering program using OpenGL. For each frame, our program computes shadow maps and estimates direct lighting with fragment shaders; indirect lighting is computed using VSLs energy only. We have decided not to take VSLs shadows into account so as to accelerate rendering process as much as possible. Consequently, some regions in the images might be sensibly over-illuminated (figure 7).

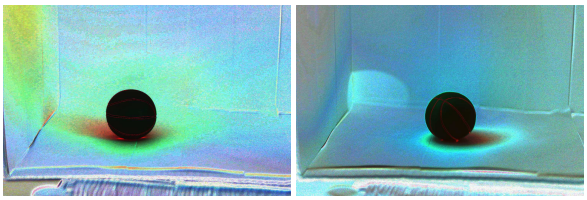
For estimating the impact of indirect shadows, we have used several photographs of the real world (figure 8). Figure 9 gives the (inverted) image differences for natural lighting (left) and using a light spot pointing the upper left corner of the box (right). The darker pixels correspond to regions where the radiance difference is high with and without the object in the scene. Note that on these photographs, many pixels are different because of indirect lighting (not only shadows). However, main (visible) indirect shadows are projected onto surfaces that are close to the considered object.

Any method usually used for virtual point light sources (instant radiosity) rendering can replace our GPU rendering system for indirect shadows, since photon tracing remains independent of rendering. Another solution could be to use ambient occlusion computations.

Our GPU rendering process requires several passes, that can benefit from deferred

Figure 7: Worst case when VSL shadows are not taken into account: the arch is illuminated by VSLs from below the floor.

Figure 8: Photographs of indirect shadows: (left) Nicolas bedroom experimentation place; (top-right) photograph with indirect natural lighting without object; (bottom-right) same lighting conditions with an object. Note the shadows on the box floor.



(a) (b)

Figure 9: Inverted difference photographs between with and without the object, the darker the image, the higher the difference: (a) with indirect illumination of natural light, coming from the window; (b) with a light spot pointing to the upper-left corner, with a bright region in the back due to residual direct lighting compensated by inter-reflexions.

shading: (i) render the scene for deferred shading; (ii) for each light source, compute the corresponding shadow map and estimate direct light contribution; (iii) for each VSL cumulate light contributions, using the G-buffer.

0.7 Technical aspects

0.7.1 Photon Streaming

Lighting simulation is achieved using photon tracing, requiring an acceleration structure. We have chosen to define a bounding box and a kd-tree for each object in the scene, so that moving one object does not require any kd-tree reconstruction. We also make the assumption that the number of object is not very high, so that no additional accelerating structure is needed for ray-object intersection.

In practice, our algorithm shoots photons carrying all the same energy, using usual photon tracing described in [Jen01]. The initial energy is defined by the light sources power; a Russian Roulette is performed for each impact for determining if the photon path should be interrupted.

0.7.2 Multi-Thread Processing

The rendering process using photon streaming can be completely separated from the rendering phase. We thus can benefit from from today computer multi-core architectures. Our program is built in two main threads. The first one performs lighting simulation while the second one achieves rendering. VPLs are shared by the two threads. They are updated continuously by the first thread, and read when necessary for each frame computation.

Synchronization is performed using mutual exclusion, so that red, green and blue values remain coherent for each VSL on each frame. However, VSLs are only read by the rendering thread and updated only on the lighting simulation thread. Without synchronization, the worst error is bounded by the energy of one photon for all the VSLs. In addition, we have made some tests without mutual exclusion and did not notice any difference.

0.7.3 Deferred Shading and Culling

First, the geometry is rendered for storing depth and normals for each frame pixel in the G-buffer. Second, we perform shadow mapping corresponding to each light source (our program handles point light sources and spot light sources). Third, each VSL is rendered as a sphere, which radius corresponds to the distance above which light contribution can be greater than $\varepsilon = 10^{-6}$.

$$\frac{\Phi_s.R(x)}{\pi.r^2} < \varepsilon \quad \Leftrightarrow \quad r > \sqrt{\frac{\Phi_s.R(x)}{\pi.\varepsilon}}$$

The sphere projection consequently updates only the image pixels under the (potential) influence of this VSL.

A bounding sphere is also associated with each object in the environment, for both kd-tree construction and frustum culling.

0.8 Results

The following results have been produced with an AMD Athlon(tm) 64 x2 dual core processor 3800+, with a Nvidia GeForce 8600GT.

We have applied photon streaming and rendering to various test scenes, 4 of them have been illustrated through this paper. Note that frame rates depend on both the number of VSLs and the image resolution, while photon streaming latency depends on the photon pool size and the number of VSLs.

Table 1 provides the description of the four scenes corresponding to the images in this paper, with the number of polygons and the number of vertices. The results given in this paper have been provided with one spot light source.

Scenes	#Polygons	#Vertices
Cornell Box	2 006	1 042
Dragon	100 010	50 022
Sponza	66 453	60 898
Sibenik	80 359	75 596

Table 1: Characteristics of four test scenes, giving the number of polygons and vertices. The Cornell Box contains a parallelepipedic box and a sphere; Dragon is also a Cornell Box containing a dragon; Sponza Atrium and Sibenik Cathedral are free models with textures.

Tables 2 and 3 provide framerates for rendering each scene using various number of VSLs, with two different image resolutions. The first value in each box corresponds to the number of images per second using deferred shading while the second value corresponds to rendering the whole scene for each VSL. Note that we did not introduce the photon pool size as a parameter since it does not influence rendering time (but the latency).

Scenes w/wo defered shading	10 VSLs	50 VSLs	100 VSLs	200 VSLs
Cornell Box	140/195	48/40	36/19	21/10
Dragon	90/50	43/9	35/5	21/2
Sponza	115/115	39/20	32/15	19/6
Sibenik	140/90	83/24	58/11	36/6

Table 2: Average fps for each scene, according to the number of VSLs, with 800x600 image resolution the first value in each box has been obtained with deferred shading while the second one has been obtained rendering the whole geometry for each rendering pass.

Scenes w/wo defered shading	10 VSLs	50 VSLs	100 VSLs	200 VSLs
Cornell Box	55/75	19/16	15/7	8/4
Dragon	45/86	19/7	15/4	8/1
Sponza	45/55	16/11	15/4	7/2
Sibenik	62/45	35/11	25/5	15/3

Table 3: Average fps for each scene, according to the number of VSLs, with 1280x1024 image resolution, with and without deferred shading.

Figure 10 provides two images inside Sibenik Cathedral. The top image shows an image rendered without indirect lighting, with a powerful spot light source. The bottom image provides the result obtained using our system, rendered at 15 frames per second.

Figure 11 provides for images of a Cornell Box, while moving the (spot) light source and the blue sphere, rendered at 20 frames per second.

0.9 Conclusions and future work

This paper presents a methods for interactively handling diffuse inter-reflections with photon streaming. We propose a system that simulates continuous photon shooting, and associates light energy to virtual points that are used during rendering as virtual elementary surface light sources (VSLs).

0.9.1 Limitations

Lighting simulation is performed continuously, and rendering each frame corresponds to taking a photograph at a given instant, with photons that can be *older* than the current frame, thus introducing some latency when moving light sources or objects.

Our method does not provides an exact solution at each frame. These errors can be reduced using usual VPLs approaches, using shadow maps for instance. Latency can be also avoided by replacing the whole photon pool at each frame, providing consequently much lower frame rates (depending on the photon pool size). However, our system is rather dedicated to rapid (pre-)visualization systems.

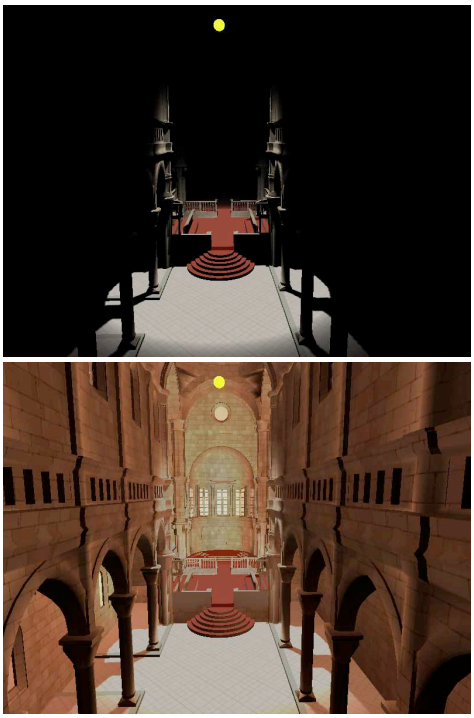


Figure 10: Two images of Sibenik Cathedral lit by a powerful spot light (top) without indirect illumination and (bottom) with indirect illumination with our rendering system.

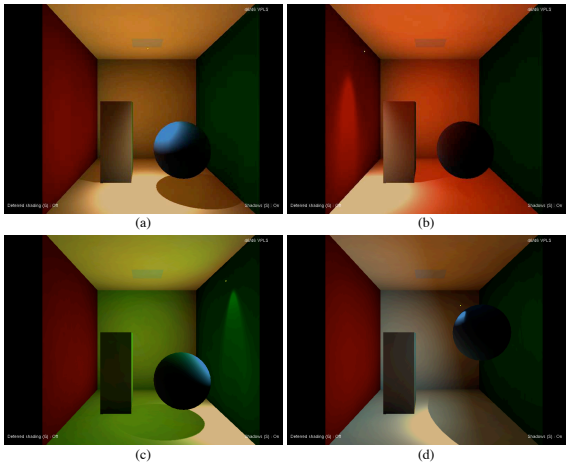


Figure 11: Several images of cornell box during interactive rendering, (a,b,c) while moving light source (d) while moving the blue sphere.

0.9.2 Future work

In the future, we aim at taking glossy inter-reflections with our system. Presently, VSLs do not provide direct information for introducing high-frequency BRDFs in the materials we use. In addition, we aim at introducing ambient occlusion and estimate the actual visual difference compared to using VSL (hard) shadows.

Bibliography

[DBMS02] DMITRIEV K., BRABEC S., MYSZKOWSKI K., SEIDEL H.-P.: Interactive global illumination using selective photon tracing. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 25–36.

[GKBP05] GAUTRON P., KŘIVÁNEK J., BOUATOUCH K., PATTANAIK S.: Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Proceedings of Eurographics Symposium on Rendering* (June 2005).

[GWS05] GÜNTHER J., WALD I., SEIDEL H.-P.: Precomputed Light Sets for Fast High Quality Global Illumination. In *ACM SIGGRAPH 2005: Sketches and Applications* (2005).

[Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[KAMJ05] KRISTENSEN A. W., AKENINE-MÖLLER T., JENSEN H. W.: Precomputed local radiance transfer for real-time lighting design. *ACM Trans. Graph. 24*, 3 (2005), 1208–1215.

[Kel97] KELLER A.: Instant radiosity. *ACM SIGGRAPH Computer Graphics* (1997), 49–56.

[KG05] KRIVANEK J., GAUTRON P.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 550–561. Member-Sumanta Pattanaik and Member-Kadi Bouatouch.

[KTHS06] KONTKANEN J., TURQUIN E., HOLZSCHUCH N., SILLION F.: Wavelet radiance transport for interactive indirect lighting. In *Rendering Techniques 2006 (Eurographics Symposium on Rendering)* (jun 2006), Heidrich W., Akenine-Möller T., (Eds.), Eurographics.

[LC04] LARSEN B. D., CHRISTENSEN N.: Simulating photon mapping for real-time applications. In *Eurographics Symposium on Rendering* (jun 2004), Henrik Wann Jensen A. K., (Ed.).

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. xx–yy.

[NPG03] NIJASURE M., PATTANAIK S., GOEL V.: Interactive global illumination in dynamic environments using commodity graphics hardware. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 450.

[pSKS02] PIKE SLOAN P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics* (2002), pp. 527–536.

[SIP06] SEGOVIA B., IEHL J.-C., PÉROCHE B.: Bidirectional Instant Radiosity. In *Proceedings of the 17th Eurographics Workshop on Rendering* (June 2006).

[SIP07] SEGOVIA B., IEHL J., PÉROCHE B.: Metropolis instant radiosity. In *EUROGRAPHICS proceedings* (2007).

[WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003).